Direct Versus Indirect Input Methods for One-Handed Touchscreen Mobile Computing

Amy K. Karlson and Benjamin B. Bederson
Human-Computer Interaction lab
Computer Science Department
University of Maryland, College Park, MD, 20721
{akk, bederson}@cs.umd.edu

ABSTRACT
In this paper, we present ThumbSpace, a software-based interaction technique that supports one-handed thumb operation of touchscreen-based mobile devices. Our goals are to provide accurate selection of all interface objects, especially small and far targets, which are traditionally difficult to interact with using the thumb. ThumbSpace is designed to provide these benefits independent of the application design. This can free designers to focus on effective presentation, as well as efficient interaction when two hands and a stylus are available. We present design tradeoffs we encountered and a comparative evaluation of ThumbSpace against peripheral hardware (indirect) and touchscreen (direct) input methods for object selection during standing and walking activities. Although ThumbSpace and another new finger-based touchscreen technique, Shift, were less accurate than the directional pad, they were preferred, in combination, to all other methods studied for interacting with 2D interfaces and small (3.6 mm) targets.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: ThumbSpace, one-handed mobile interaction

INTRODUCTION
Not only have mobile phones become a ubiquitous social accessory, but rapid advances in their processing power and storage capacity have transformed them into "smartphones": feature-rich, Internet-enabled mini PCs. While the most widespread styles in circulation feature the classic combination of numeric keypad and non-touchscreen display, Apple's announcement of the iPhone earlier this year has experts predicting an explosion of touchscreen devices [20]. As the hardware features of these devices evolve, exemplified by the iPhone's breakout multi-touch touchscreen, so will opportunities to explore a wider range of interaction methods. The persisting constraints on design will be the finite availability of users' visual, physical, and mental resources during mobile activities [13]. In our work, we have focused on the limitation that mobile users often have only one hand available to use a device.

Interfaces that accommodate single-handed interaction can offer a significant benefit by freeing one hand for the physical and intellectual demands of mobile tasks [16]. Surveys [11] confirm that users would generally prefer to use touchscreens with one hand when possible, but today's hardware and software designs typically offer little support. Styli are often required for touchscreen interaction because their interfaces can be composed of targets are too small [17] or ill-positioned to be hit reliably with the thumb [11].

One option is to build interfaces that explicitly ensure all targets are thumb sized and within thumb reach [12]. But a "lowest common denominator" approach to interface design is unlikely to catch on because screen real estate is a precious resource for small devices, and placing limits on visual expressivity can hurt the design in other ways. As an example, increasing object sizes to accommodate thumbs means fewer objects are displayed per screen, so more screens are required to present a given amount of data. This can slow information access when two hands *are* available. This observation, and the fact that existing mobile UI toolkits tend to support only small, stylus-oriented widget palettes, have led us to consider an alternative strategy.

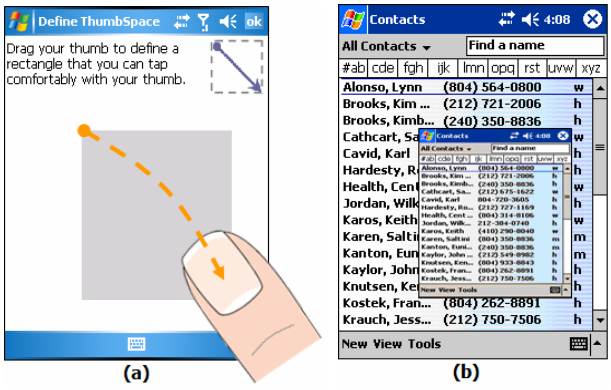

Figure 1: (a) Defining ThumbSpace. (b) ThumbSpace as a Radar View.

ThumbSpace aims to address both the reach and accuracy problems that users experience when operating touchscreens with thumbs. Its design is inspired by the substantial body of research that has focused on the challenge that

distance plays in large display interaction. The ThumbSpace design reflects our interpretation of how solutions for accessing objects out of arm's reach on large displays can be adapted to the problem of accessing objects out of thumb reach on handheld devices. Conceptually, ThumbSpace serves as an absolute touchpad superimposed on the display, to which all screen objects are mapped (Figure 1b). Reach limitations are addressed by allowing users to personalize the size and placement of ThumbSpace (Figure 1a), thereby accommodating individual differences in hand preference, geometry, motion range, grip, and use scenario. Not only does ThumbSpace support access to all display objects within an accessible portion of the screen, but we apply dynamic visual feedback and selection tuning to alleviate high error rates that are typical in hitting small targets with big thumbs. Finally, ThumbSpace is only available upon a user action and so is unobtrusive when not enabled.

At the core of our design is an assumption that users of touchscreen devices will be compelled to touch the interface, even when not using a stylus. But problems of high visual demand, targeting accuracy and finger occlusion suggest peripheral hardware solutions might be more appropriate. For example, Blackberry devices have shown how thumbwheels can be used quite effectively. After presenting the ThumbSpace design, we describe a study which compares the efficacy of ThumbSpace to alternative touchscreen techniques and to peripheral hardware input methods to better understand these approaches. ThumbSpace was introduced in [10]. This paper extends that work by describing the design trade-offs and iterations in significantly more detail, coupled with a new controlled study.

RELATED WORK

One handed interaction

Research in one-handed mobile interaction has largely focused on specific user tasks, such as media play [18], text entry [24], and application navigation [12]. Karlson et al. [11] looked more generally at human factors requirements for one-handed use of mobile devices, including situational and task preferences for hand use as well as biomechanical limitations of thumbs. They found widespread interest in single-handed operation of personal devices, but that current designs, especially of touchscreen devices, do not accommodate one-handed scenarios well. Their results also show that the areas of a device users are comfortable interacting with vary by user and device size. ThumbSpace is specifically designed to addresses these findings.

Many current touchscreen interfaces consist of widgets similar in size and function to those featured on a desktop PC. While acceptable for interaction with a 1mm stylus tip, research suggests touchscreen targets smaller than 9.6 mm [17] can result in unacceptably high error rates when accessed one handed with the thumb. Recently, Vogel and Baudisch [23] developed the Shift technique as an improvement over Sears and Shneiderman's offset cursor [21], a highly regarded approach to precision touchscreen targeting for over 15 years. The offset cursor couples a selection cursor positioned off the tip of a user's finger with a stabilization algorithm to achieve character-level selection accuracy. The downside to the offset cursor is that users have to aim *below* the intended target. Shift instead allows users to aim directly for the target, and after a variable delay, displays a portal view (callout) of the screen area covered by the user's finger. The callout includes a crosshair cursor to show the position of finger contact, which users can adjust before lifting their finger to perform selection. The delay function that determines when to show the callout is proportional to the size of the target under the finger - short for small targets, and longer for large targets. The result is that Shift is only shown when users are likely to need it, and does not interfere with selection otherwise.

While Shift holds great potential for one-handed selection of targets within reach of the thumb, further investigation is necessary to understand whether pixel-level selection is appropriate under mobile conditions, and whether Shift works equally well for objects along the perimeter of the screen, which occur frequently in today's designs. ThumbSpace, on the other hand, is expected to support targets at edges just as well as those away from the edges. More importantly, Shift was designed for two-handed index finger operation of mobile devices, and so does not address the limitations of thumb reach that ThumbSpace does.

Reaching Distant Objects

ThumbSpace draws its inspiration from table-top and wall-sized displays, which both confront problems with out-of-reach interface objects. A general problem in large display interaction is that the increase in real estate also increases the average distance between on-screen objects. Unfortunately, Fitts' Law dictates that increasing travel distance without a commensurate increase in target size will increase access time. Solutions have thus typically focused on 1) decreasing movement distance to targets and/or 2) increasing target sizes.

Improving target acquisition for mouse-based interaction has often involved clever manipulation of the control-display (CD) ratio. Slowing mouse movement over interaction targets (Semantic Pointing [6]), jumping the cursor to the nearest target (Object Pointing [8]), and predicting the user's intended target (Delphian desktop [2]) are three such examples. The drawback of these techniques is that their effectiveness decreases as the number of nearby objects increases. Other approaches in smart cursor control make targets easier to hit by increasing the cursor size, such as area cursor [9] and Bubble Cursor [7]. Unfortunately, these techniques are not directly applicable to touchscreen interaction; touching the screen directly means a 1:1 correspondence between motor and display movement, so there is no CD ratio or cursor to manipulate.

Direct screen interaction with fingers or pens is common in tablet, mobile, and wall computing. Techniques to improve object access speed in these arenas have focused on minimizing the movement distance to targets. However, most research that focuses on icon placement or selection [3, 5, 14] is inappropriate for PDA interfaces because drag and

drop and object placement are used much less frequently than interactions such as tapping buttons, links, and check boxes. Another approach has been to provide a nearby miniaturized version of the display, or Radar View [15], that can be manipulated directly. However, both the Radar View and the pen-based extension to the Bubble Cursor, the Bubble Radar [1], again focus on object placement tasks, rather than general application interaction.

THUMBSPACE DESIGN

Our goal with ThumbSpace has been to develop an interaction strategy whereby rich touchscreen interfaces can be effectively controlled with a thumb, without sacrificing the expressiveness of information presentation or the efficiency of navigation when two hands are available.

Given individual variation in thumb agility [11], hand size, strength, and usage scenario, the first principle of ThumbSpace is to support each user's most comfortable and stable grip. Each user therefore defines her own ThumbSpace - a region of the touchscreen surface that she considers easy to reach and interact within. The user may redefine the position and size of ThumbSpace anytime after an initial configuration step in which she drags her thumb along a diagonal to define the upper right and lower left corners of a rectangular region (Figure 1a). All thumb interaction then occurs within this personalized ThumbSpace, which remains fixed across all applications.

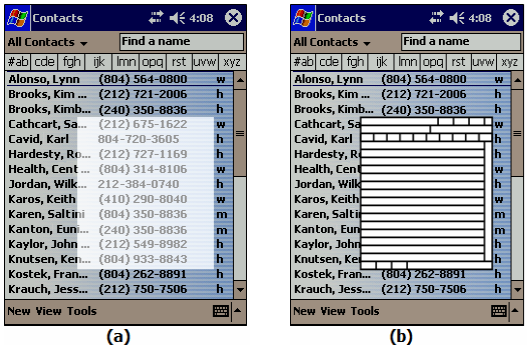

Figure 2: (a) The initial ThumbSpace representation. (b) One possible partitioning of the ThumbSpace into proxies.

To support access to all interaction targets within the confines of the ThumbSpace, the region behaves as Radar View. Consider, for example, a Radar View applied to the Windows Mobile Contacts application in Figure 1b. We see that a simple version of this approach would have several problems: 1) the Radar View representation occludes a large number of DisplaySpace objects; 2) the Radar View proxies are unreadable; 3) the detailed Radar View representation contributes to visual clutter; and 4) the Radar View objects are far too small to hit reliably with the thumb.

Initial Design

Our first ThumbSpace design addressed problems (1-3) by avoiding the use of a miniature representation entirely. Instead, we offered only a whitewashed region to visually suggest where the user should focus her attention (Figure 2a). In this design, ThumbSpace overlaid the application at all times. Although no miniature representation was shown, ThumbSpace behaved as a Radar View by honoring an input mapping between the ThumbSpace and the original display. ThumbSpace was partitioned so that each object in the original display was associated with a sub-region (proxy) in the ThumbSpace; tapping a proxy in ThumbSpace selected the associated object in the original display. Assuming the ThumbSpace represents a linear scaling of the original display (e.g., Figure 1b), the partition of ThumbSpace into proxies would be that of Figure 2b.

Using ThumbSpace

The final challenge (re: (4) above) in using a miniature representation as an interaction platform is that the proxies will likely be too small to hit reliably with a thumb. This is true even when the users can see the representation (as in Figure 1b), but our initial ThumbSpace design introduced further uncertainty because it failed to provide visual cues for how its sub-regions mapped to display objects. ThumbSpace managed these uncertainties by providing a visual feedback loop during object selection.

Object selection with ThumbSpace is performed in three phases: *aim*, *adjust*, and *lift*. The *aim* phase depends on users forming a mental model of the mapping between the ThumbSpace proxies and display objects, with the natural assumption being that the ThumbSpace represents a linearly scaled version of the original display. Based on this model, the user touches the sub-region of the ThumbSpace she believes best corresponds to the intended target (Figure 3a). In the *aim* phase, ThumbSpace can be likened to an absolute touchpad - if the user guesses correctly, ThumbSpace provides direct access to objects that would otherwise be difficult (e.g., too small) or impossible (e.g., out of reach) to hit directly with her thumb.

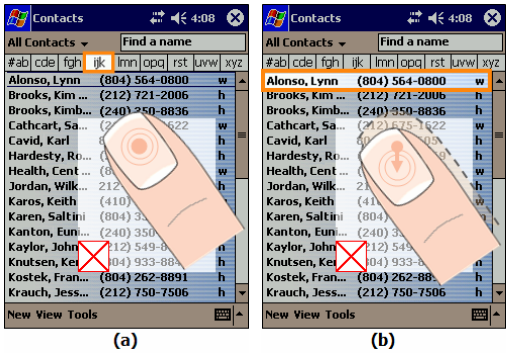

Figure 3: Selecting objects with ThumbSpace. Assuming the user wants to select the first name in the Contacts list, she first (a) aims at the ThumbSpace proxy for 'Alonso'; (b) the initial ThumbSpace contact point maps to 'ijk' so the user adjusts selection by dragging her thumb down. The user confirms the selection by lifting her thumb.

Once the thumb touches a ThumbSpace proxy, the associated display object is visually identified with an object cursor, depicted by a thick orange border. The user then enters

the *adjust* phase of selection. During the *adjust* phase, ThumbSpace acts like a relative touchpad for controlling the object cursor. If the user rolls or drags her thumb more than a fixed number of pixels up, down, left, or right, the object cursor animates to the closest display object in the direction of movement (Figure 3b). In the *adjust* phase, ThumbSpace interaction is similar to Object Pointing [8] in desktop environments, which ignores the white space between interface objects, and instead jumps the mouse pointer to the nearest object in the direction of movement once the pointer leaves an object's border. Our *adjust* strategy differs slightly from Object Pointing because the *adjust* threshold is independent of the display object sizes.

Finally, the user confirms the selection by lifting her thumb. This manner of object selection is inspired by the lift-off strategy for touchscreen object selection developed by Potter [19], which allows users to visually confirm and adjust a selection before committing to the action. Our initial ThumbSpace design allowed users to cancel a selection, by dragging her thumb over a red X, which would appear throughout the *adjust* phase in the corner furthest from the point of initial thumb contact. A video demonstration of ThumbSpace definition and interaction can be viewed at http://www.youtube.com/user/thumbspace.

Evaluation of Initial ThumbSpace Design

To understand the efficacy and usability characteristics of our initial design, we conducted a quantitative study to compare ThumbSpace to direct thumb interaction for accessing targets. Sixteen participants (8 male, 8 female) used both techniques (direct touch and ThumbSpace) to select objects of varying size (small: 20 px vs. large: 40 px), density (sparse vs. dense), and position (near vs. far).

As we report in [10], results showed that overall error rates were comparable between the two techniques, but that participants were consistently slower using ThumbSpace. Even so, ThumbSpace made clear progress toward two of our design goals: (1) "decouple target size from thumb size": users were as effective at selecting small targets as large targets, evident from the fact that neither speed nor error data were affected by target size; (2) "improve selection for targets that are out of thumb reach": users accessed far targets more accurately using ThumbSpace than direct interaction. Finally, users felt ThumbSpace held promise, giving it relatively high ratings on a 7-point scale for task execution satisfaction (5.1), fun (5.4) and learnability (5.4), and the majority of users indicated that ThumbSpace would be preferable to direct interaction after sufficient practice. These results encouraged us that a strategic redesign could have a strong impact on closing the performance gap between ThumbSpace and direct thumb input.

THUMBSPACE REDESIGN

Because ThumbSpace was inspired by the challenge of accessing all areas of a touchscreen while using a device with one hand, our initial design inadvertently favored targets that were out of thumb reach. This is because the ThumbSpace itself interfered with near targets, making them more difficult to hit. But our ideal solution should make hard tasks easy, without negatively impacting tasks that are already easy. Our response is to allow users to trigger ThumbSpace on demand. In this way touchscreen operation will be no worse than direct interaction, but for targets that are hard to access with the thumb (e.g., small or far), user have the option of using ThumbSpace. We chose a hardware trigger because it is reliable and unambiguous. Users might choose any of the re-mappable hardware buttons on their own device, but for our test platform, the Cingular 8525, we used the center of the directional navigation pad (e.g., "enter") for its positional convenience.

Figure 4: Final ThumbSpace design. (a) A calendar. (b) Pressing the middle button of the directional navigation pad launches a mini representation within the user's predefined ThumbSpace. (c) If the user wants to hit June 26th, she aims for the 26th in the mini representation. (d) Upon touch down, the display object associated with the proxy she hits is highlighted at the same time the ThumbSpace disappears. The user may drag her thumb to adjust the cursor, or lift to perform the selection.

The second goal of our redesign was to reduce object selection time. Observations and interaction logs both suggested that our study participants did not make "good" guesses in the *aim* stage of object selection (Figure 3a), and so spent considerable time making adjustments (Figure 3b). To reduce adjustments, we needed to support users in making better guesses in the first place, which for us meant providing better visual cues. We reasoned that the most common use of ThumbSpace would be to access targets that are out of thumb reach. Since ThumbSpace is by definition *within* thumb reach, it would not interfere with such targets, and so could be used to provide the necessary visual cues.

Our new design uses a faithful miniature of the display as the ThumbSpace (Figure 4b). However after the *aim* phase, the ThumbSpace disappears to avoid occlusion during se-

lection adjustment. In our study implementation, we did not implement a cancel feature, although cancel could be provided in the future, for example, by moving the selection cursor off the screen in any direction.

Our final design adjustment was to constrain each user's ThumbSpace rectangle to the same aspect ratio as the display itself. In our initial design, users were allowed to define a ThumbSpace with arbitrary dimensions, which we believed made the mental transformation between ThumbSpace and the display more challenging than necessary. Matching aspect ratios can help users locate ThumbSpace proxies quickly, because visual cues are reinforced by spatial memory, rather than requiring mental gymnastics.

DIRECT TOUCH VS. PERIPHERAL INPUT HARDWARE

Our fundamental assumption in developing ThumbSpace was that for interfaces designed for stylus input, tapping targets directly with the thumb would be faster and more natural than using peripheral input hardware, such as scroll wheels or directional navigation pads. But the wild popularity of RIM's Blackberry, which uses a thumbwheel almost exclusively for object selection, suggests either that this assumption is incorrect, or that the Blackberry software and interaction experience have been optimized for the thumbwheel. While it is certainly true that thoughtful hardware/software coupling can offer powerful advantages in task efficiency and user satisfaction (which then would not be true of the arbitrary touchscreen interfaces we wanted to support), there are several key reasons peripheral hardware input might be more appropriate than touchscreen interaction for one-handed mobile computing.

Stable One-Handed Operation

Limiting information navigation and selection activities to a single hardware component that can be comfortably accessed with a one-handed grip (as the scroll wheel does) not only frees the other hand for additional activities that arise in a mobile setting (such as carrying bags), but also means users can operate the device without changing their grip, even when accessing information across a variety of applications. If the hardware is strategically located so that the grip is stable, users can perform information tasks while mobile without risk of dropping the device, which otherwise diverts physical and attention resources that are more appropriately directed toward navigating the environment.

Occlusion

In addition, using hardware on the periphery of the device keeps the screen free from visual interference. This is in contrast to interaction methods that involve touching the screen directly, which must contend with finger and hand occlusion. Since today's devices are equipped with standard resistive touchscreens, they can accept only one input point at a time; if the screen is touched in more than one place, only the average point is registered as the input point. When using a stylus, this characteristic is not typically a problem since a stylus has a small tip for precise targeting, and offers a thin, low profile extension to the hand that keeps bulky fingers away from the screen. Since fingers have much larger contact areas than styli, users must aim at targets with the center of their finger pad, which can then completely cover small targets. In this case, the average input point may fall outside the target bounds and generate a selection error.

Reduced Visual and Mental Demand

Object selection with direct touch involves a unified process of continual visual monitoring and physical adjustment. This can be a detriment for mobile settings as users may need to divide their visual attention between the device and the environment. Input via peripheral hardware, on the other hand, has two separable components: *aim* and *selection*; aim positions a cursor, and selection activates the object under the cursor. While the aim phase *may* require continual visual monitoring, as when using a mouse, it does not have to. Scroll wheels and directional pads convey abstract commands, rather than position information, to move the cursor from one selectable object to the next. With these methods, the cursor position depends only on the number and type of commands issued, so constant visual monitoring is unnecessary if the user preplans the command sequence to translate the cursor to the desired target.

Another advantage of peripheral hardware is that the state of the cursor is maintained between commands. This allows the aim phase to be interleaved with myriad stimuli that compete for the user's visual and mental attention in mobile computing. Furthermore, cursor stability supports high precision selection. Finally, by lowering the visual, mental, and physical demands of device interaction, peripheral hardware solutions effectively push device activities to the background, which is a positive influence on the safety of mobile users and others surrounding them.

THUMBSPACE USER STUDY

Countering the numerous advantages peripheral hardware input enjoys over touchscreen interaction for one-handed mobile device operation, there are challenges as well. Cursors must move serially from object to object. Each of these movements takes time away from the ultimate goal of selecting a target. This characteristic gives direct touch a potential advantage in average interaction speed. Of course both speed *and* accuracy influence user satisfaction, so our goal is to understand the relative advantages of peripheral hardware vs. touchscreen input methods for task speed, accuracy, and satisfaction for one handed device use.

Independent Variables

Input Methods

In selecting input methods to include in our investigation, we wanted to compare ThumbSpace to the common alternatives used with today's devices. For peripheral hardware we chose the scroll wheel (*ScrollWheel*) and the directional navigation pad (*DPad*) – ScrollWheel for being the distinguishing feature of the non-touchscreen Blackberry devices and DPad because nearly every cell phone has one. It was clear that for touchscreen interaction, we would compare ThumbSpace to direct touch (*DirectTouch*), since that is how users operate touchscreens one-handed today. We

assumed ThumbSpace would offer users more accurate targeting at the expense of speed, so also chose a technique that specifically addresses targeting accuracy for fingers on touchscreen. Recently, Vogel and Baudisch [23] showed users made fewer errors using their Shift technique over direct touch for hitting targets ≤ 2.9 mm, but not ≥ 4.3 mm. Given the possibility that Shift would help users in hitting the small targets (3.6 mm) used in our study, we included it as a more competitive variant of DirectTouch.

Mobility

Many previous studies have established the negative impact movement has on mental demand and task performance (e.g., [16]). If our assumption that the hardware input methods are more stable and require less mental, visual, and physical demand than the touchscreen methods, then increased activity would be expected to degrade performance more when using touchscreen methods than hardware methods. To understand this relationship, we studied users performing tasks while both standing and walking. During the walking condition, users chose a comfortable walking pace along a 19' x 7.5' tape figure eight.

Target Sizes

Our initial study of ThumbSpace [10] confirmed that user performance was independent of target size, so here we chose only a single target size of 20x20 pixels (3.6 mm^2), representative of standard Windows Mobile widgets (e.g., checkboxes: 15 px, buttons: 21 px, text boxes: 19 px).

Figure 5: (a) Input regions (dark="hard to reach", light="easy to reach"). (b) The Cingular 8525.

Tasks

Tasks were based on selection activities that would typically be performed with two hands using a stylus. Since our goal was to understand appropriate one-handed interaction techniques for rich interfaces we selected a two-dimensional input space of arbitrarily placed objects. Potential targets were placed within a 6x8 grid of 40x40 px^2 cells. We chose this number of targets because it is comparable to the number of targets offered in some standard applications that have 2D layouts, such as the month view of Windows Mobile Calendar (52 objects). For analysis purposes, we partitioned the targets into a 3x4 grid of regions, 4 targets per region (Figure 5a). Regions were labeled "easy to reach" (light gray) or "hard to reach" (dark gray) based on the majority opinion of study participants.

Because the targets were smaller than their assigned cells (20 vs. 40 px), the target for each trial was placed in the center the designated cell. All other distracter targets were randomly assigned one of two sizes (20 px and 13 px, with probabilities 0.2 and 0.8 respectively), and were positioned at random locations within their cells. The randomized locations were used to create the illusion of a non-uniform input space, while the variable sized objects increased the percentage of the background displayed. The targets were placed on a map background because we expected context to be useful during use of the Shift technique.

Tasks were presented as a dialog that indicated the trial target and the action to take to begin the task (Figure 6a). To help users distinguish the goal target from among others, its center was colored yellow. The message was always placed at the center of the user's personal ThumbSpace, unless it overlapped the target, and then was placed either above or below the target. For ScrollWheel, users pressed the scroll wheel to begin; for DPad and ThumbSpace, users pressed the center of the directional pad to begin; for DirectTouch and Shift, users tapped the dialog to begin.

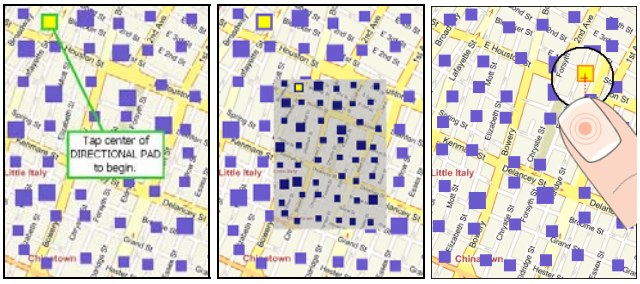

Figure 6: Study screen shots. (a) Task instruction; ThumbSpace (b) and Shift (c) representations.

A rectangular orange cursor was used as the input focus. For ScrollWheel and DPad, the cursor started at the upper left target, since this is a typical home position for cursors in today's interfaces. For DirectTouch, Shift and ThumbSpace, the cursor only appeared once the user had touched the screen to perform a selection. When a target was selected, the input cursor would animate toward the midpoint of the target and vanish to provide visual feedback, while an audio sound also indicated trial success or failure.

Hypotheses

Based on intuition and previous research suggesting users are faster and more satisfied when interacting with a compact region of the device [11], we had the following hypotheses regarding the touchscreen interaction methods.

(H1) DirectTouch would be faster than either Shift or ThumbSpace; (H2) ThumbSpace and Shift would be more accurate than DirectTouch; (H3) ThumbSpace would be faster than Shift in the "hard to reach" regions (Figure 5a); and (H4) Shift would be faster than ThumbSpace in "easy to reach" regions.

In comparing peripheral hardware input methods to the touchscreen methods, we hypothesized: (H5) DPad and ScrollWheel would be more accurate than DirectTouch, Shift, and ThumbSpace; (H6) Shift and ThumbSpace would be faster on average than both DPad and Scroll-

Wheel; (H7) DPad and ScrollWheel would be less impacted by walking than the touchscreen methods.

Implementation and Apparatus

As a real-world input system, ThumbSpace will need to cooperate with a PDA's operating system in order to capture and reinterpret thumb events. However, to first establish its viability, we implemented ThumbSpace as an input handler to custom applications written in C# (.NET Compact Framework 2.0) for Windows Mobile Pocket PCs using the PocketPiccolo.NET graphics toolkit [4]. The software ran on a 400 MHz Cingular 8525 PocketPC phone (Figure 5a) with a 43.2 mm x 57.6 mm display area of 240 x 320 pixels. The device was chosen because it was is the only device available that had all the hardware components we studied (a touchscreen, directional navigation pad, and scroll wheel), thus avoiding a potential confound.

Since touchscreens recognize only a single average point from a finger touch, that point is not only hard to predict, but also unstable, due to continual updates as the soft finger tip deforms on the flat screen surface. This problem is well known to make pixel-level targeting difficult, so various approaches have been used to stabilize finger input [21, 23]. In our work we take the approach used in Shift [23], transforming each input point by applying a recursive dynamic filter to stabilize pointer movement during slow corrective movements [22]. We found cutoff frequencies of 5 and 20 Hz interpolated between 54 and 144 mm/s worked well for both ThumbSpace and Shift. Given the small size of our targets, Shift was configured to escalate (display the callout) as soon as the finger touched the screen, but we did not correct for users' perceived contact points as in [23].

Several techniques for touchscreen object selection have been reported and studied in the literature. Two of the most common approaches are land-on and take-off, for which the first or last input points, respectively, are taken as the point of selection. We tested both techniques in pilots with mixed results. Since targets are completely hidden once the user's finger has landed, users cannot adjust the selection in an informed way. We reasoned that land-on reflected the user's best guess at the target, and chose that approach for the DirectTouch condition. For consistency with the other touchscreen-based input techniques, we filtered the input points using the same parameters as ThumbSpace, and registered a change in selection only if the user moved her thumb at least 20 pixels away from the point of contact, in effect stabilizing the selection.

Method

The study was a 5 (*Input*: DPad, ScrollWheel, DirectTouch, Shift, ThumbSpace) x 2 (*Mobility*: standing, walking) x 12 (*Region*) x 4 (*Position*) repeated measures within-subjects factorial design. Presentation of *Input* and *Mobility* were counterbalanced across participants, and the 48 region x position trials were randomized within blocks.

Dependent variables collected included task time, error rate, satisfaction ratings, and interface preference rankings.

Participants

Twelve right handed volunteers (8 male, 4 female) ranging in age from 21 to 31 (μ = 26) were recruited via fliers posted in the Department of Computer Science. Participants received $15 for 1.5 hours of their time.

Procedure

Before each block of trials, the study administrator explained and demonstrated the input method that would be used. Participants were instructed to select each target as quickly as possible without sacrificing accuracy. For walking conditions, participants were asked to walk at a comfortable pace along the figure-8 during all trials.

Participants then assumed a standing position or began walking, and began the practice trials. For DirectTouch, DPad and ScrollWheel, users performed 20 random practice trials the first time they saw the input method, and 10 the second time. Since Shift and ThumbSpace were new to users, they performed 40 random practice trials the first time they encountered the input method, and 20 the second time. After the practice trials, users performed the 48 timed tasks. After each block, participants filled out a short subjective questionnaire about the *Input* x *Mobility* condition. Users proceeded in this manner for all 10 *Input* x *Mobility* blocks. A final block of trials offered users the option of using DirectTouch or ThumbSpace, to gather information about when, if ever, users would choose to launch ThumbSpace based on the position of the target.

Participants then completed a "usability" phase, which provided the opportunity to use all input methods with a realistic interface. For this phase users remained standing as the study software presented 10 tasks for each of the 5 Input conditions (DPad, ScrollWheel, DirectTouch, Shift, and ThumbSpace) in that order (roughly familiar to unfamiliar) for a Windows Mobile Calendar interface (Figure 4). Following the usability phase, users ranked the input methods from 1= "favorite" to 5= "least favorite" by target type (2: easy-to-reach and hard-to-reach), mobility condition (2: standing and walking), and expected overall preference once sufficient practice and expertise had been achieved. This last question included an option for using ThumbSpace when desired, and Shift otherwise.

STUDY RESULTS

Task Times

Task time was measured form the onset of the trial (when the scroll wheel or center of the directional pad was released, or the user's finger was lifted from the task dialog) to the completion of the trial (when the scroll wheel or center of the directional pad was pressed, or the user's finger was lifted from the screen). Task times for each region were determined by averaging the region's 4 position trials. Trials with selection errors were excluded from the aggregation and Huynh-Feldt corrections were used when sphericity did not hold.

A 5 (*Input*: DPad v. ScrollWheel v. DirectTouch v. Shift v. ThumbSpace) x 2 (*Mobility*: standing v. walking) x 12 (*Region*: 1-12) repeated measures Analysis of Variance

(RM-ANOVA) was carried out on the average task time data. Main effects of input $F(3, 18.1)=55.9$, $p< .001$, and region $F(11,66)=44.4$, $p<.001$, were observed. In addition, an interaction of input x region $F(44,264)=32.2$, $p<.001$, was also observed.

On average, DirectTouch was significantly faster (874 ms) than all other interaction methods; ScrollWheel was significantly slower (3324 ms) than the others, while Shift, ThumbSpace, and DPad did not differ significantly from one another (Figure 7a). These results support H1, but not H6, since DPad was as fast as Shift and ThumbSpace.

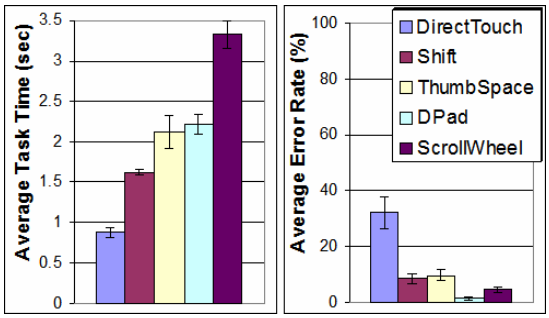

Figure 7: Average task times (a) and error rate (b).

Average task times increased consistently with region number, where region 1 was the fastest and region 12 was the slowest. While this seems at first unintuitive, the result can be understood by considering the input x region effect, whereby DPad and ScrollWheel task times generally increased by region, while DirectTouch, Shift and ThumbSpace remained constant across regions (Figure 8), resulting in an average overall increase in task time by region. Paired t-tests revealed Shift was significantly faster than ThumbSpace in regions (5,6,7,8,10,11) largely confirming H3. However, H4 failed to be supported since ThumbSpace was never faster than Shift.

Note that task time patterns for DPad and ScrollWheel clearly illustrate the 2D and linear target access approaches of each. For DPad and ScrollWheel, region 1 is fastest because the cursor starts there. For DPad, the time taken to access targets in the other regions is linear in the number of 2D steps required, creating equivalence groups: (2,4), (3,5,7), (6,8,10) and (9,11). With ScrollWheel, accessing a target in a region requires traveling through all lower numbered regions, hence the increasing time by region. It is important, therefore, to be clear that average task times for DPad and ScrollWheel will depend on the total number of interface objects and the relative access frequencies of each.

Error Rate

A 5 (*Input*) x 2 (*Mobility*) x 12 (*Region*) RM-ANOVA was performed the average percent error data. Main effects of input $F(1.3,14.9)=21.2$, $p<.001$, mobility $F(1,11)=5.1$, $p=.004$, and an interaction of input x region $F(44,484)=2.6$, $p<.001$, were found.

On average, users were slightly more accurate while standing than walking (10% v. 12% error). With 32% error, DirectTouch was significantly less accurate than any of the other input methods (Figure 7b), which supports H2. DPad (1% error), on the other hand, was significantly more accurate than ScrollWheel (5% error), Shift (8% error) or ThumbSpace (10% error), which were otherwise indistinguishable from one another. These results fail to confirm H5, since we expected that ScrollWheel to be as accurate as DPad. User comments indicated that ScrollWheel's error rate was likely due to accidental movement when the scroll wheel was pressed to perform selection. Different scroll wheels may vary in their susceptibility to this type of error.

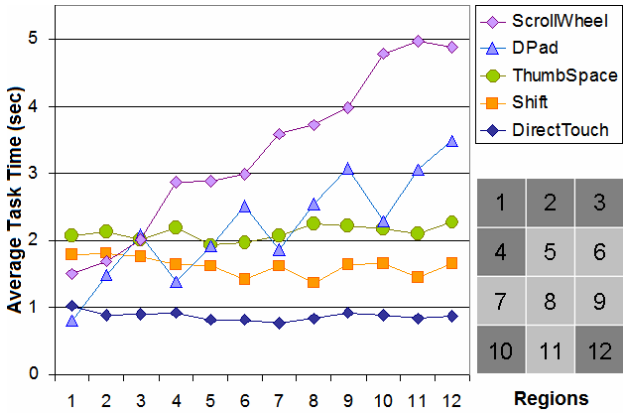

Figure 8: Task times by region for each input.

Examining the input x region data, we found the accuracies of DPad, ScrollWheel, and Shift were generally stable across regions. ThumbSpace showed a comparable error rate to Shift and ScrollWheel across regions 2-8, but significantly higher accuracy in region 1 and significantly lower accuracy in regions (9, 11, 12) than either Shift, DPad or ScrollWheel, all at $p <= .05$ level. User accuracy with DirectTouch was the most variable among input methods, proving most stable within regions 6-10 (30-33% error), highest in region 1 (20% error), but particularly low in region 3 (50% error).

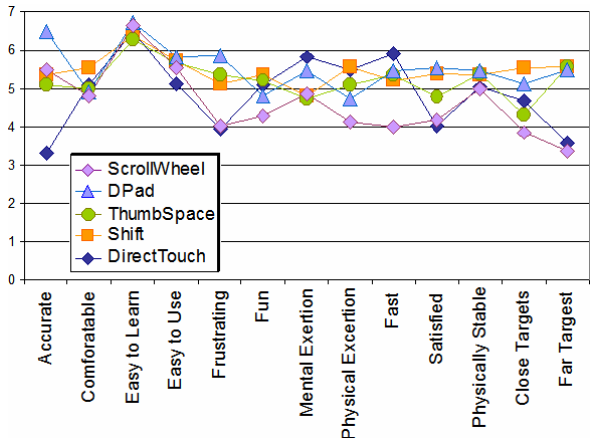

Figure 9: Satisfaction ratings for each input.

Satisfaction

We performed a 5 (*Input*) x 2 (*Mobility*) x 13 (*Rating*) RM-ANOVA on participant satisfaction ratings, selected from a

7 point scale, (1=low, 7=high satisfaction). A main effect of rating $F(12,120)=7.9$, $p<.001$ and an interaction between input and rating $F(48,480)=4.3$, $p<.001$ were found.

As shown in Figure 9, DPad, Shift, and ThumbSpace scores were on average fairly high (e.g. majority ≥ 5), and were generally comparable across ratings. Where they differed most, DPad was considered more accurate and ThumbSpace was less preferred for "close targets" than the other two. DirectTouch ratings were similar to DPad, Shift, and ThumbSpace, except it was considered less accurate, more frustrating, less satisfying, and less useful for "far targets" than the other three. Finally, ScrollWheel generally received lower ratings than the other input methods.

Preference

Based on their experience, we asked participants to provide an absolute ranking of the 5 input methods, plus a sixth option of using a Shift+ThumbSpace combination, from 1=Best to 6=Worst for the majority of device interaction. The Shift+ThumbSpace was most popular on average (2.1), followed by Shift (2.6), ThumbSpace tied with DPad (3.3), then DirectTouch (3.8), and ScrollWheel (5.4). Although Shift received the most top rankings (4 users), it was closely followed by DPad and Shift+ThumbSpace (3 users). In fact, 75% of participants ranked Shift+ThumbSpace as their first or second choice, as opposed to only 42% for Shift, 33% for DPad, and 25% for ThumbSpace.

DISCUSSION

With respect to our broad questions about the relative value of peripheral hardware vs. touchscreen interaction for mobile, one-handed computing, we found little evidence that one approach should be recommended over the others. While DPad enjoyed surprisingly high satisfaction ratings, was relatively fast, and was the most accurate selection method, it ranked below 2 of the touchscreen methods with respect to absolute preference, and was the only method that generated unsolicited comments about hand fatigue. The fact that ScrollWheel was the least preferred input method is almost certainly related to the large number of targets, which made average selection time slow, and the 2D left-to-right, top-to-bottom cursor movement, which may have seemed an inefficient route to target objects. However, this means only that ScrollWheel may not be suitable for dense 2D interfaces, and not that ScrollWheel is an ineffective input method in general. On the contrary, Blackberry devices are a great example of how thoughtful, complementary software designs can render scroll wheels both effective and enjoyable.

Of the touchscreen methods studied, it is clear that direct thumb input with today's technology is still too imprecise for reliable access to small targets (e.g., ≤ 9mm). While nearest-target selection strategies may be used to improve direct thumb use in sparse displays, densely populated displays must still rely on precise targeting. Shift and ThumbSpace, however, proved to be effective software solutions to improving user accuracy for hitting small (3.6 mm) targets, which are common in PocketPC interfaces, and are otherwise easy to hit with two-handed stylus interaction. In terms of average speed, accuracy, and satisfaction ratings, the two techniques were effectively indistinguishable. In considering different regions of the device, however, speed, accuracy, and satisfaction ratings were higher for Shift in "easy-to-reach" regions, which are precisely those for which we assumed ThumbSpace would not be required. (hence our ThumbSpace redesign to allow voluntary launching). In "easy-to-reach" regions the primary concern is imprecise targeting due to thumb occlusion, which is the problem for which Shift is particularly suited.

Given these data, it is unsurprising that Shift was preferred to ThumbSpace as a method users would favor for the *majority* of their device interaction. However overall preference for using Shift *with* ThumbSpace suggests occasions when ThumbSpace offered advantages over Shift, and users were indeed able to access targets in the upper left corner more reliably using ThumbSpace than Shift. With its 2.8" screen, the device we used is one of the smaller touchscreens on the market, and so region 1 may have been the only one that was truly hard to reach. Many recent generations of PocketPC, as well as the much anticipated iPhone, have 3.5" displays that are up to 32% longer and wider than the device studied. In their investigation of the effect device size has on one-handed thumb movement, Karlson et al. [11] reported users generally have difficulty reaching all areas of the screen when using devices this large (e.g. the PDA in their study). The personalizability of ThumbSpace in terms of size, position, and use occasion, mean it can flexibly and unobtrusively accommodate variations in hand size, agility, and device dimension. That both Shift and ThumbSpace were considered Fast, Accurate, Easy to Learn, Easy to Use make them a very promising combination for generalized one-handed touchscreen operation.

Finally, the fact that Shift+ThumbSpace ranked at the top of the preference ratings despite neither having an advantage in speed or accuracy over DPad bolsters our intuition that users prefer pointing at targets directly for selecting on-screen objects rather than using indirect access methods. One user's comment "I'm not sure I am actually faster using Shift (vs. DPad), but I *feel* that I am" offers some insight into why this could be the case. Before Shift and ThumbSpace, users did not have a competitive touchscreen alternative to direct thumb interaction. Since most devices offer a directional navigation pad as a standard hardware feature, users can now decide for themselves which methods best support their personal speed/accuracy tolerances.

CONCLUSION

In this paper we presented the iterative design of ThumbSpace, a generalized approach for operating rich touchscreen-based interfaces with a thumb. ThumbSpace borrows solutions to reaching problems from large display interaction, and refines them for use on small displays; a Radar View offers ballistic thumb targeting of screen objects, regardless of their size and location, and modified Object Pointing allows users to refine their selection for high accuracy.

We compared the use of ThumbSpace to two peripheral hardware input methods, and two touchscreen based methods, for accessing small (3.6 mm) targets in a 2D layout while both standing and walking. Our results showed the relative advantages of the input methods were independent of activity level. The directional pad offered a good balance between speed and accuracy, but the touchscreen approaches were generally preferred. While thumb touch alone generated unacceptably high error rates, Shift and ThumbSpace still allow users to point at targets with their thumbs, but offered users significantly higher input accuracy. Shift primarily addresses occlusion and precision problems when the thumb is much larger than the intended target, and ThumbSpace addresses reach limitations when holding a device in one hand. Used in combination, Shift can offers high accuracy thumb access to objects that are within reach, while ThumbSpace provides the same benefits for targets out of thumb reach.

ACKNOWLEDGMENTS

Many thanks to Patrick Baudisch for helpful brainstorms and Dan Vogel for the recursive dynamic filter.

REFERENCES

1. Aliakseyeu, D., Subramanian, S., Gutwin, C. and Nacenta, M., Bubble Radar: Efficient Pen-Based Interaction. *Proc. AVI '06*, (2006), ACM Press, 19-26.
2. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K. and Kishino, F., Predictive interaction using the delphian desktop. *Proc. UIST '05*, (2005), ACM Press, 133-141.
3. Baudisch, P., Cutrell, E., et al., Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. *Proc. Interact '03*, (2003), 57-64.
4. Bederson, B.B., Meyer, J. and Good, L., Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. *Proc. UIST '00*, (2000), 171-180.
5. Bezerianos, A. and Balakrishnan, R., The vacuum: facilitating the manipulation of distant objects. *Proc. CHI '05*, (2005), ACM Press, 361-370.
6. Blanch, R., Guiard, Y. and Beaudouin-Lafon, M., Semantic pointing: improving target acquisition with control-display ratio adaptation. *Proc. CHI '04*, (2004), ACM Press, 519-526.
7. Grossman, T. and Balakrishnan, R., The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proc. CHI '05*, (2005), ACM Press, 281-290.
8. Guiard, Y., Blanch, R. and Beaudouin-Lafon, M., Object pointing: a complement to bitmap pointing in GUIs. *Proc. Graphics Interface '04*, (2004), Canadian Human-Computer Communications Society, 9-16.
9. Kabbash, P. and Buxton, W., The "Prince" technique: Fitts' Law and selection using area cursors. *Proc. CHI '95*, (1995), ACM Press, 273-279.
10. Karlson, A.K. and Bederson, B.B., ThumbSpace: Generalized One Handed Input for Touchscreen-Based Mobile Devices. *To appear in Proc. INTERACT '07*, (2007), Springer.
11. Karlson, A.K., Bederson, B.B. and Contreras-Vidal, J.L. Understanding One Handed Use of Mobile Devices. in Lumsden, J. ed. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, Idea Group, 2007, (in press).
12. Karlson, A.K., Bederson, B.B. and SanGiovanni, J., AppLens and LaunchTile: two designs for one-handed thumb use on small devices. *Proc. CHI '05*, (2005), ACM Press, 201-210.
13. Kristoffersen, S. and Ljungberg, F., Making place to make it work: empirical exploration of HCI for mobile CSCW. *Proc. GROUP*, (1999), ACM Press, 276-285.
14. Mountaz, H., Throwing models for large displays. *Proc. HCI '03*, (2003), British HCI Group, 73-77.
15. Nacenta, M.A., Aliakseyeu, D., Subramanian, S. and Gutwin, C., A comparison of techniques for multi-display reaching. *Proc. CHI '05*, (2005), ACM Press, 371-380.
16. Oulasvirta, A., Tamminen, S., Roto, V. and Kuorelahti, J., Interaction in 4-sec. bursts: the fragmented nature of attentional resources in mobile HCI. *Proc. CHI '05*, (2005), ACM Press, 919-928.
17. Parhi, P., Karlson, A.K. and Bederson, B.B., Target Size Study for One-Handed Thumb Use on Small Touchscreen Devices. *Proc. Mobile HCI '06*, (2006), ACM Press, 203-210.
18. Pirhonen, P., Brewster, S.A. and Holguin, C., Gestural and audio metaphors as a means of control in mobile devices. *Proc. CHI '02*, (2002), ACM Press, 291-298.
19. Potter, R.L., Weldon, L.J. and Shneiderman, B., Improving the accuracy of touch screens: an experimental evaluation of three strategies. *Proc. CHI*, (1988), 27-32.
20. Robinson, S. Apple iPhone: Catalyst for Capacitive Touchscreen-Only Phones to Balloon to 115 Million Units within Two Years, Strategy Analytics, 2007.
21. Sears, A. and Shneiderman, B. High-precision touchscreens: design strategies and comparisons with a mouse. *Intl. J. of Man-Machine Stud.*, 34 (4). 593-613.
22. Vogel, D. and Balakrishnan, R. Distant freehand pointing and clicking on very large, high resolution displays. *Proc. UIST '05*. 33-42.
23. Vogel, D. and Baudisch, P., Shift: A technique for operating pen-based interfaces using touch. To appear in *Proc. CHI '07*, (2007).
24. Wigdor, D. and Balakrishnan, R., TiltText: using tilt for text input to mobile phones. *Proc. UIST '03*, (2003), ACM Press, 81-90.